UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/731,060	12/07/2000	Edward Colles Nevill	550-192	1332

23117 7590 08/20/2007

NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
ZHEN, LI B	

ART UNIT	PAPER NUMBER
2194	

MAIL DATE	DELIVERY MODE
08/20/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

mn

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

MAILED

AUG 2 0 2007

Technology Center 2100

Application Number: 09/731,060
Filing Date: December 07, 2000
Appellant(s): NEVILL ET AL.

John R. Lastova (Reg. No. 33,149)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 25 April 2007 appealing from the Office action mailed 21 September 2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,937,193	Evoy	08-1999
6,374,286	Gee et al.	04-2002

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

- Claims 1 – 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,937,193 to Evoy in view of U.S. Patent No. 6,374,286 to Gee et al.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation

under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 1 – 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,937,193 to Evoy [cited in the previous office action] in view of U.S. Patent No. 6,374,286 to Gee et al. [hereinafter Gee].

As to claim 1, Evoy teaches the invention substantially as claimed including an apparatus [computer system 10, Fig. 1; col. 3, lines 29 – 42] for processing data operable to execute operations specified in a stream of program instructions [converting platform-independent instructions to be executed by a processor into corresponding native instructions for the processor; col. 4, lines 9 – 20 and col. 3, lines 42 – 60], the apparatus comprising:

(i) a hardware based instruction execution unit operable to execute program instructions [a translation circuit 50 coupled to system data bus 24 is utilized to receive 8-bit Java bytecodes and output corresponding 32-bit native instructions directly to processor 40 for execution; col. 4, lines 52 – 62]; and

(ii) a software based instruction execution unit [software interpreter; col. 5, lines 57 – 67] operable to execute program instructions [interpret the unmapped bytecode via a software interpreter; col. 5, lines 57 – 67, col. 6, lines 1 – 8, col. 7, lines 8 – 16]; wherein

(iii) program instructions to be executed are sent to the hardware based execution unit [a platform-independent instruction (here a Java bytecode) is fetched from the memory; col. 10, lines 45 – 57] for execution [Translate Code routine 200, for translation state machine 153; col. 32 – 45]; and

(iv) program instructions received by the hardware based execution unit for which execution is not supported by the hardware based execution unit are forwarded to the software based execution unit for execution [If no corresponding native instruction exists, table 51 outputs an exception signal, which notifies processor 40 that software interpretation of the bytecode may be required; col. 7, lines 8 – 17; col. 5, lines 57 – 67; col. 6, lines 1 – 8; col. 9, lines 54 – 65; col. 11, lines 23 – 37] with control being returned to the hardware based execution unit for a next program instruction to be executed [selecting next bytecode; col. 7, lines 8 – 16; col. 11, lines 6 – 15].

As to scheduling support logic, Evoy schedules instructions to be executed by the processor and dispatching the next instruction to be executed [After execution of a native instruction when in the platform-independent mode, processor 40 increments its address counter to the next instruction, which has the effect of selecting the next bytecode provided to byte select multiplexer 56; col. 7, lines 17 – 28]. However, Evoy does not specifically disclose scheduling operation for managing scheduling between threads or tasks irrespective of whether a preceding program instruction was executed by the hardware based execution unit or the software based execution unit.

However, Gee teaches hardware based execution unit [JEM processor 100, Fig. 1; col. 8, line 58 – col. 9, line 5], software based execution unit [processor opcodes will

trap to software to resolve the class reference and replace the null CSA ptr; col. 13, lines 50 – 56], and scheduling support logic to generate a scheduling signal for triggering [partition interval timer 1712 is used to signal the completion of a partition time slice and return to JVM0 operation, which, as previously mentioned, checks for system events and schedules the next partition; col. 28, lines 43 – 52] a scheduling operation to be performed between program instructions [thread scheduling software; col. 21, lines 25 - 57] for managing scheduling between threads or tasks [In the inventive JEM processor, a priority-based scheduler which conforms to the above-described rules dispatches (makes ready to execute) the highest priority thread from the set of all runnable, or "ready", threads; col. 21, lines 43 – 58] irrespective of whether a preceding program instruction was executed by the hardware based execution unit or the software based execution unit [decision concerning which thread to run at any particular moment is made in accordance with a scheduling policy, col. 21, lines 25 – 60; Examiner notes that Gee discloses scheduling of threads according to a scheduling policy and not based on whether a preceding program instruction was executed by the hardware based execution unit or the software based execution unit].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Gee and Evoy because Gee's teachings provide a conventional JAVA scheduling policy is defined in the JAVA Language Specification that implements a preemptive, priority-based scheduling policy [col. 21, lines 27 – 36 of Gee] and allows threads with the highest priority to execute first in a real-time embedded system [col. 21, lines 47 – 50 of Gee].

As to claim 16, this is a method claim that corresponds to apparatus claim 1; note the rejection to claim 1 above, which also meet this method claim.

As to claim 2, Evoy teaches the scheduling support logic includes a counter with a value [address counter; col. 7, lines 17 – 27] that is changed in response to a program instruction sent to the hardware based execution unit [After execution of a native instruction when in the platform-independent mode, processor 40 increments its address counter to the next instruction; col. 7, lines 17 – 27].

As to claim 3, Evoy teaches the counter triggers generation of said scheduling signal when a predetermined count value is reached [END reserved code; col. 10, line 65 – col. 11, line 6].

As to claim 4, Evoy teaches the counter may be programmed to start from a user programmable start value [translation vector base address register 172 stores the starting address of the first table; col. 9, lines 27 – 47].

As to claim 5, Evoy teaches the counter counts up to said predetermined value [processor 40 increments its address counter to the next instruction; col. 7, lines 15 – 28].

As to claim 6, Evoy as modified by Gee teaches the counter counts down to said predetermined value [partition interval timer 1712; col. 28, lines 43 – 52 of Gee;

examiner notes that a timer is a counter that can count up or down to a predetermined value. Therefore, the interval timer corresponds to the timer.].

As to claim 7, Evoy as modified by Gee teaches a debug operation is triggered by the scheduling signal [col. 11, lines 22 – 38 of Evoy and col. 26, lines 17 – 30 of Gee].

As to claim 8, Evoy does not teach timer logic. However, Gee teaches timer logic operable to generate a timer signal indicative of a time since a last scheduling operation [partition switch time-out watchdog timer 1716; col. 29, lines 12 – 29 of Gee]. It would have been obvious to a person of ordinary skilled in the art at the time the invention was made to combine the teachings of Gee and Evoy because Gee's teachings prevent one thread from taking over the system by refusing to relinquish control [col. 23, lines 18 – 30 of Gee].

As to claim 9, Evoy as modified by Gee teaches the scheduling signal is combined with said timer signal to trigger said scheduling operation [partition switch time-out watchdog timer 1716 is used to unconditionally terminate partition execution when the partition interval timer interrupt (1804) is not acknowledged; col. 29, lines 12 – 29 of Gee].

As to claim 10, Evoy as modified by Gee teaches a scheduling operation is triggered upon generation of said scheduling signal after said timer signal has reached

a predetermined value indicating a predetermined period time since a last scheduling operation has expired [col. 29, lines 12 – 29 of Gee].

As to claim 11, Evoy teaches a processor core operable to execute operations as specified by instructions of a first instruction set [col. 4, lines 38 – 44 and col. 7, lines 50 – 61].

As to claim 12, Evoy teaches the hardware based instruction execution unit includes an instruction translator [a hardware-implemented translation circuit; col. 4, lines 9 – 20] operable to translate instructions of a second instruction set into translator output signals corresponding to instructions of the first instruction set [if a corresponding native instruction to the selected bytecode exists, table 51 outputs it over data lines 54a and directly to processor 40; col. 7, lines 8 – 16].

As to claim 13, Evoy teaches (i) at least one instruction of the second instruction set specifies a multi-step operation that requires a plurality of operations that may be specified by instructions of the first instruction set in order to be performed by the processor core [translation state machine 153 handles exception conditions by testing for an EXC code. Upon receipt of this code, control is passed to block 212 where the translation process is halted, the REQ signal is released, and an interrupt is sent to processor 140; col. 11, lines 22 – 38]; and

(ii) the instruction translator is operable to generate a sequence of translator output signals to control the processor core to perform the multi-step operation [multiple

IMM codes may be required, with separate processing, to handle immediate data of different lengths; col. 11, line 36 – col. 12, line 5].

As to claim 14, Evoy teaches the software based execution unit is a software based interpreter [software interpreter; col. 5, lines 57 – 67].

As to claim 15, Evoy teaches the program instructions are Java Virtual Machine instructions [col. 4, lines 52 – 62].

(10) Response to Argument

Appellant argues in substance that:

(1) Thus, contrary to the Examiner's assertion, 7:8-16 of Evoy does not disclose the claimed feature where control is returned to the hardware based execution unit for a next program instruction to be executed. When Evoy's processor is in native mode, the address counter points to native instructions and incrementing the address counter selects the next native instruction for execution. Control is not returned to the non-native, platform independent mode once software interpretation has been invoked. [pp. 8 – 11];

(2) Evoy never teaches returning control the hardware-based execution unit, corresponding to the translation circuit 50 used in the platform-independent mode of Evoy, after a bytecode has been forwarded to the software based execution unit (the software interpreter) for execution. [p. 10];

(3) As clearly stated in claim feature (v), the hardware based execution unit includes scheduling support logic that generates "a scheduling signal for triggering a scheduling operation to be performed...." It is improper for the Examiner to add to or remove words from the claims. [p. 12];

(4) A simple timer-based scheduling approach may suffer from the disadvantage that scheduling operations may be inappropriately triggered at points part-way through the software interpretation of a complex program instruction in a manner that could cause a loss of data integrity should an inappropriate context switch occur. [pp. 12 – 13];

(5) Gee's conventional scheduling policy is just another example of a known scheduling policy. Gee does not disclose a scheduling policy like that claimed. The issue of managing scheduling "irrespective of whether a preceding program instruction was executed by the hardware based execution unit or the software based execution unit" does not arise in Gee because Gee does not disclose both the hardware-based and software-based execution units. [pp. 13 – 14];

(6) Even if one included Gee's conventional highest priority thread scheduling in Evoy, the result would still suffer from the possibility that the highest priority thread may be scheduled part way through software interpretation of a complex non-native program instruction. According to claims 1 and 16, the program instructions are sent to the hardware based execution unit and forwarded to the software based execution unit, with control being returned to the hardware based execution unit, to ensure that a scheduling

operation will not be performed in the middle of execution of the Java bytecode, which often requires the execution of several native instructions. [pp. 15 – 18];

(7) The cited passage of Evoy discloses passing control to process a new instruction in the event that the new instruction is not an END reserve code. This is different from generating a scheduling signal when a counter has reached a predetermined count value as specified by claim 3. [p. 19].

Examiner respectfully traverses Appellant's arguments:

As to arguments (1) and (2), it is noted that claims only require returning to the hardware execution unit after instruction execution and do not require not switching from one mode of execution to another mode. Evoy discloses that after execution of the bytecode via software interpretation, the selection of the next bytecode is received from the multiplexer [col. 7, lines 15 – 28]. The multiplexer [element 56, Fig. 2] is a component of the translation unit [hardware execution unit] and provides the next bytecode to the processor [i.e. col. 7, lines 1 – 17]. In order for the multiplexer to provide the next program instruction to be executed (bytecode) to the processor, control has to be returned to the hardware based execution unit because the multiplexer is part of the translation unit and provides the next bytecode for execution.

As to point (3), examiner agrees with appellant that the claims do not require a software scheduling code to manage scheduling between program instructions irrespective of whether a preceding program instruction was executed by the hardware based execution unit or the software based execution unit. Instead, feature (v) of the

claims only require the generation of a scheduling signal for triggering a scheduling operation to be performed. The claims are silent as to what component receives the generated scheduling signal and performs the scheduling operation. For the sake of completeness, it was noted that based on the specification, the scheduling operation is performed by scheduling code that is identified as software [see Fig. 10, element 76; p. 26, line 30 – p. 27, line 3 and p. 27, lines 16 – 26].

As to point (4), examiner notes that appellant's specification and claims also disclose controlling scheduling operations using a timer based approach [Figs. 11 and 12, claims 8 – 10]. However, the claims do not recite specific features that prevent scheduling operations from being inappropriately triggered at points part-way through the software interpretation. Therefore, the recited claims fail to distinguish appellant's invention from the timer-based scheduling of the prior art.

As to argument (5), Gee discloses hardware based execution unit [JEM processor 100, Fig. 1; col. 8, line 58 – col. 9, line 5] and software based execution unit [software to resolve the class reference and replace the null CSA ptr; col. 13, lines 50 – 56]. Gee discloses the use of software to execute instructions [resolve references for opcodes, i.e. col. 13, line 50 – col. 14, line 62] and control is returned to the JVM0 after execution completion [i.e. col. 28, lines 45 – 52]. The JEM processor contains a priority-based scheduler that dispatches (makes ready to execute) the highest priority thread from the set of all runnable, or "ready", threads [col. 21, lines 43 – 58] based on a scheduling policy [col. 21, lines 25 – 60]. Therefore, Gee discloses scheduling of threads according to a scheduling policy and not based on whether a preceding

program instruction was executed by the hardware based execution unit or the software based execution unit.

As to argument (6), examiner disagrees with appellant allegation that the prior art "may" schedule a thread for execution part way through software interpretation of a complex non-native program instruction. It is noted that the claims do not specifically disclose ensuring that a scheduling operation will not be performed in the middle of execution of a Java bytecode. Appellant's asserts that the feature of returning control to the hardware based execution unit ensures that a scheduling operation will not be performed in the middle of execution of the Java bytecode. Evoy teaches that after execution of a native instruction when in the platform-independent mode, processor increments its address counter to the next instruction, which has the effect of selecting the next bytecode provided to byte select multiplexer over system data bus [col. 7, lines 15 – 28]. The byte select multiplexer [element 56, Fig. 2] is included in the translation circuit [element 50, Fig. 2; i.e. col. 5, lines 10 – 23] and provides the next bytecode to the processor [i.e. col. 7, lines 1 – 17]. Evoy discloses that the instruction is executed to completion (not interrupted) and control is returned to the to byte select multiplexer of the translation circuit [hardware based execution unit] to select the next bytecode for execution. Since Evoy also discloses returning control to the hardware based execution unit after the software based execution unit executes the instruction, Evoy also provides the feature that ensures a scheduling operation will not be performed in the middle of execution of the Java bytecode.

As to argument (7), the END reserve code in Evoy triggers the determination of whether the current source address is equal to the source end address. If source address equals the source end address, translation is completed and the processor is notified that it is done by deasserting the REQ signal and passing an interrupt request to the processor. When the address counter has reached a predetermined value [source address is equal to the source end address], a scheduling signal is generated [interrupt request to the processor].

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Li B. Zhen

August 13, 2007

Conferees:

Eddie C. Lee

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER

WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER

William D. Thomson

Li B. Zhen